Theoretical Computer Science 38 (1985) 99–116
North-Holland

FORMAL SYSTEMS FOR JOIN DEPENDENCIES

Catriel BEERI and Moshe Y. VARDI*
Department of Computer Science, The Hebrew University of Jerusalem, Jerusalem 91904, Israel

Communicated by J.D. Ullman
Received April 1981
Revised November 1983

Abstract. We investigate whether a sound and complete formal system for join dependencies can be found. We present a system that is sound and complete for tuple generating dependencies and is strong enough to derive join dependencies from join dependencies using only generalized join dependencies in the derivation. We also present a system that sound and complete for tuple generating dependencies and is complete for extended join dependencies (which are a special case of generalized join dependencies). Finally, we construct a Gentzen-style system that is sound and complete for join dependencies. The last two systems have unbounded inference rules.

Key words. Database, relational model, join dependency, implication problem, formal system.

1. Introduction

The most widely studied design method for relational database schemes is the *decomposition* method [13]. A *join dependency* [1, 16] is a semantic specification by the database designer of a lossless decomposition. There are also other classes of dependencies, all of which are semantic specifications of some kind.

A problem of utmost importance for database design theory is the implication problem for join dependencies: does a set of join dependencies imply another join dependency. That is, given that certain decompositions are lossless, can we tell that another decomposition is also lossless. An implication testing algorithm, the 'chase', was constructed by Maier et al. [15]. This algorithm has an exponential worst-case running time. Moreover, there is probably no polynomial algorithm, since deciding if a decomposition is lossless when given two join dependencies is NP-hard [10].

The chase enables us to test implications of join dependencies. In the process of database design, it is useful to know all of the dependencies implied by a given set. There is, however, no way to find this set using the chase, without exhaustively enumerating the set of all possible dependencies. Consequently, we are led towards finding a *formal system* for join dependencies; a formal system enables us to *derive*

* Research partially supported by Grant 1849/79 of the U.S.A.-Israel Binational Science Foundation.
Current address: Center for Study of Language and Information, Ventura Hall, Stanford University, Stanford, CA 94305, U.S.A.

new join dependencies from given ones. Formal systems for dependencies have attracted a lot of interest in the last few years, since the introduction of a formal system for functional dependencies by Armstrong [2].

Since the implication problem for join dependencies is recursively solvable, the formal system consisting of the inference rule

$$d_1, \ldots, d_k \vdash d \quad \text{if } \{d_1, \ldots, d_k\} \models d$$

is sound and complete. Nevertheless, there is an interest in finding an 'elegant' formal system, one that has a small number of simple axioms and inference rules. A typical example is the propositional calculus, which is a formal system for the recursive set of tautologies of propositional logic.

Up to now all attempts to develop a sound and complete formal systems for join dependencies have failed. (The formal system of [8, 17, 18] has been shown to be complete only for deriving first order hierarchical decompositions [8, 18]). Thus, attention has shifted to finding classes of dependencies that include join dependencies as a special case and for which there is a sound and complete formal system. One such is the class of *total tuple generating dependencies* [9], for which a formal system was developed in [11].

Sciore [17] has defined the class of *generalized join dependencies*, which lies strictly between the classes of tuple generating dependencies and join dependencies. He constructed a formal system which can derive join dependencies from join dependencies by derivations that consist of generalized join dependencies. In Section 3.1 we improve Sciore's result. We show that the formal system for tgd's in [11] has a similar property: it can derive join dependencies from join dependencies by derivations that consist of full generalized join dependencies. Our treatment is much simpler than that of [17].

A formal system can be viewed as a computing mechanism for generating consequences. A class of dependencies may not have a sound and complete formal system, because no sound formal system for the class can have enough computing power to generate all consequences. There are two possible solutions to the problem. One can consider a larger class of dependencies, or one can consider a stronger notion of formal system. In Section 3.2 we combine both approaches. We define the class of *extended join dependencies*, which lies strictly between the classes of join dependencies and generalized join dependencies. We then develop a sound and complete system for these dependencies. This system differs from most systems in the literature in having an *unbounded* rule, i.e., a rule with an unbounded number of premises.

Finally, by extending the notion of formal system even further, we manage to get a formal system for join dependencies. All of the formal systems in the literature of dependency theory are Hilbert-style, in which a derivation is a sequences of dependencies. In contrast, in Gentzen-style systems a derivation is a sequence of *sequents*, which are formal statements of implication [14]. In Section 4 we present a Gentzen-style system with an unbounded rule and prove that it is sound and complete for join dependencies.

2. Basic definitions

2.1. Attributes and relations

Attributes are symbols taken from a given finite set U called the *universe*. All sets of attributes are subsets of U. We use the letters $A, B, C, \ldots$ to denote single attributes, and $Q, R, \ldots$ to denote sets of attributes. We do not distinguish between the attribute A and the set $\{A\}$. The union of X and Y is denoted by XY, and the complement of X in U is denoted by $\bar{X}$. An *attribute set collection* (asc) is a set of subsets of U whose union is U. We use $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{S}, \ldots$ to denote asc's.

With each attribute A is associated an infinite set called its *domain*, denoted $\mathrm{DOM}(A)$, such that $\mathrm{DOM}(A) \cap \mathrm{DOM}(B) = \emptyset$, for $A \neq B$. Let $\mathrm{Dom} = \bigcup_{A \in U} \mathrm{DOM}(A)$. For a set $X \subseteq U$, an *X-value* is a mapping $w: X \to \mathrm{Dom}$ such that $w(A) \in \mathrm{DOM}(A)$ for all $A \in X$. A *relation on X* is a finite set of X-values. We use the letters $t, u, \ldots$ to denote values, and $I, J, \ldots$ to denote relations. A *tuple* is a U-value. An arbitrary relation, unless explicitly stated otherwise, is a relation on U.

2.2. Operations on relations

We use two operations of the relational algebra [12]. For an X-value w and a set $Y \subseteq X$ we denote the restriction of w to Y by $w[Y]$. Let I be a relation on X. The *projection* of I on Y, denoted $\pi_Y(I)$, is $\pi_Y(I) = \{w[Y]: w \in I\}$.

Let $I_1, \ldots, I_k$ be relations on $X_1, \ldots, X_k$, respectively. The *join* of $I_1, \ldots, I_k$, denoted $I_1 * \cdots * I_k$, is

$$I_1 * \cdots * I_k = \{w: w \text{ is an } X_1 \ldots X_k\text{-value s.t. } w[X_j] \in I_j, \text{ for } 1 \leqslant j \leqslant k\}.$$

With each asc $\boldsymbol{R} = (R_1, \ldots, R_k\}$ we associate a *project-join expression*

$$m_{\boldsymbol{R}} = \pi_{R_1} * \cdots * \pi_{R_k}.$$

This expression defines a mapping from relations to relations as follows. Let I be a relation, then

$$m_{\boldsymbol{R}}(I) = \{w: w \text{ is a tuple s.t. for all } R \in \boldsymbol{R} \text{ there is a tuple } u \in I \text{ s.t. } w[R] = u[R]\}.$$

Project-join expressions were studied in [7, 18]. We mention some properties:

(1) $I \subseteq m_{\boldsymbol{R}}(I)$.

(2) $m_{\boldsymbol{R}}(m_{\boldsymbol{R}}(I)) = m_{\boldsymbol{R}}(I)$.

(3) $I \subseteq I$ entails $m_{\boldsymbol{R}}(I) \subseteq m_{\boldsymbol{R}}(J)$.

A *valuation* is a mapping $h: \mathrm{Dom} \to \mathrm{Dom}$, such that $a \in \mathrm{DOM}(A)$ entails $h(a) \in \mathrm{DOM}(A)$ for all $a \in \mathrm{Dom}$. The valuation h can be extended to tuples and relations as follows: Let w be a tuple. Then $h(w) = h \circ w$ ($\circ$ denotes functional composition). Let I be a relation. Then $h(I) = \{h(w): w \in I\}$.

A *tableau* [3] is a pair $T=\langle w, I\rangle$, where w is a tuple and I is a relation, such that $w[A]\in \pi_A(I)$ for all $A\in U$. T defines an operation on relations as follows:

$$T(J)=\{h(w): h \text{ is a valuation s.t. } h(I)\subseteq J\}.$$

That is, $T(J)$ is the set of images of w under all valuations that map every tuple of I to some tuple of J. Observe that $J\subseteq T(J)$.

Example 1. Let $U=AB$. Let I be the relation

A	B
$a0$	$b1$
$a1$	$b1$
$a1$	$b0$

Let w be the tuple

A	B
$a0$	$b0$

Now $T=\langle w, I\rangle$ is a tableau. Let J be the relation

A	B
$a0$	$b0$
$a1$	$b0$
$a1$	$b1$
$a2$	$b1$
$a2$	$b3$

$T(J)$ is the relation

A	B
$a0$	$b0$
$a1$	$b0$
$a1$	$b1$
$a2$	$b1$
$a2$	$b3$
$a0$	$b1$
$a1$	$b3$
$a2$	$b0$

Clearly, the values in a tableau serve as formal variables, and therefore can be renamed, if done consistently.

Lemma 2.1 ([3]). *Let $\langle w, I\rangle$ be a tableau, and let h be a one-to-one valuation. Then, for every relation J, $\langle w, I\rangle(J) = \langle h(w), h(I)\rangle(J)$.*

We now show how to construct a tableau that defines the same mapping as a project-join expression. Let $\boldsymbol{R} = \{R_1, \ldots, R_k\}$. $T_{\boldsymbol{R}}$ is a tableau $\langle w, I\rangle$, where w is an arbitrary tuple, $I = \{w_1, \ldots, w_k\}$, $w_i[R_i] = w[R_i]$, and $w_i[A]$ is a value that has a unique occurrence in I, for all $A \in \bar{R}_i$.

Lemma 2.2 ([3]). *For all relations I, $m_{\boldsymbol{R}}(I) = T_{\boldsymbol{R}}(I)$.*

We say that $T_{\boldsymbol{R}}$ *represents* $m_{\boldsymbol{R}}$.

Consider now the following problem. Given a tableau $\langle w, I\rangle$, under what condition is it a tableau $T_{\boldsymbol{R}}$ which represents a project-join expression $m_{\boldsymbol{R}}$. Let $u \in I$. Then $u[A]$ is *repeated* in I if there is another tuple $v \in I$ such that $u[A] = v[A]$. $u[X]$ is nonrepeated in I if for no $A \in X$ is $u[A]$ repeated in I.

Lemma 2.3 ([18]). *Let $\langle w, I\rangle$ be a tableau. Then it represents a project-join expression $m_{\boldsymbol{R}}$ if and only if:*

(1) *For all $A \in U$, at most one A-value is repeated in I.*

(2) *If $u[A]$ is repeated in I, then $u[A] = w[A]$.*

Example 2. Let $U = ABC$. Let $\boldsymbol{R} = \{AB, AC, BC\}$. $T_{\boldsymbol{R}}$ is the tableau $\langle w, I\rangle$:

	A	B	C
w:	$a0$	$b0$	$c0$
	$a0$	$b0$	$c1$
I:	$a0$	$b1$	$c0$
	$a1$	$b0$	$c0$

Let T_1, T_2 be tableaux. We say that T_1 is *covered* by T_2, denoted $T_1 \leqslant T_2$, if $T_1(I) \subseteq T_2(I)$, for every relation I.

Lemma 2.4 ([3]). *Let $\langle u, I\rangle$ and $\langle v, J\rangle$ be tableaux. The following conditions are equivalent:*

(1) $\langle u, I\rangle \leqslant \langle v, J\rangle$.

(2) $u \in \langle v, J\rangle(I)$.

(3) *There is a valuation h on J such that $h(J) \subseteq I$ and $h(v) = u$.*

Put otherwise, $\langle u, I\rangle \leqslant \langle v, J\rangle$ if and only if there are a valuation h and a relation I' such that $u = h(v)$ and $I = h(J) \cup I'$. Searching for the appropriate h can, however, be quite difficult, since testing covering of tableau is NP-complete [3, 10].

Example 1 (*continued*). Let T' be $\langle w, J\rangle$, where J is

A	B
$a0$	$b3$
$a3$	$b3$
$a3$	$b1$
$a1$	$b5$
$a5$	$b5$
$a5$	$b1$
$a1$	$b7$
$a7$	$b7$
$a7$	$b0$

To show that $T \leqslant T'$, we compute $T'(I)$ and get

A	B
$a0$	$b1$
$a1$	$b1$
$a1$	$b0$
$a0$	$b0$

Now $w \in T'(I)$, so $T \leqslant T'$. However, $T(J)$ is

A	B
$a0$	$b3$
$a3$	$b3$
$a3$	$b1$
$a1$	$b5$
$a5$	$b5$
$a5$	$b1$
$a1$	$b7$
$a7$	$b7$
$a7$	$b0$
$a0$	$b1$
$a1$	$b1$
$a1$	$b0$

Now $w \notin T(J)$, so $T \not\leqslant T'$.

2.3. Dependencies

For any given application only a subset of all possible relations is of interest. This subset is defined by constraints which are to be satisfied by the relation of

interest. A class of constraints that was extensively studied is the class of dependencies.

A *join dependency* (jd) is a statement $*[\boldsymbol{R}]$ or $*[R_1, \ldots, R_k]$, for an asc $\boldsymbol{R} = \{R_1, \ldots, R_k\}$. It is satisfied by a relation I if $I = m_{\boldsymbol{R}}(I)$.[1] Intuitively, $*[\boldsymbol{R}]$ means that I can be represented by the projections $\pi_{R_1}(I), \ldots, \pi_{R_k}(I)$ without loss of information. A *tuple generating dependency* (tgd) is a tableau $\langle w, J\rangle$.[2] It is satisfied by a relation I if $I = \langle w, J\rangle(I)$. Intuitively, $\langle w, J\rangle$ means that if some tuples, fulfilling certain conditions, exist in the relation, then another tuple must also exist in the relation. Thus, we can view a tableau both as an operation on relations and as a constraint. By Lemma 2.2, every jd is equivalent to some tgd; hence, we can view a tableaux of the form $T_{\boldsymbol{R}}$ as a jd, and say that it represents $*[\boldsymbol{R}]$. The class of jd's is denoted by JD, and the class of tgd's is denoted by TGD. Clearly, JD $\subset$ TGD.[3] Note that, for a given universe U, TGD is an infinite set, while JD is a finite set.

A dependency is *trivial* if it is satisfied by every relation.

Lemma 2.5 ([9])

(1) *The jd* $*[\boldsymbol{R}]$ *is trivial if and only if* $U \in \boldsymbol{R}$.

(2) *The tgd* $\langle w, I\rangle$ *is trivial if and only if* $w \in I$.

For a set of dependencies D we denote by SAT(D) the set of relations that satisfy all dependencies in D. D *implies* a dependency d, denoted $D \models d$, if SAT(D) $\subseteq$ SAT(d). That is, if d is satisfied by every relation that satisfies all dependencies in D. The *implication problem* is to decide for a given set of dependencies D and a dependency d whether $D \models d$. An algorithm that tests implication of tgd's, the chase, was developed in [9], generalizing the algorithm for jd's in [15].

In the sequel, D denotes a finite set of dependencies, and d and d' denote single dependencies.

Intuitively, to test whether $D \models \langle w, I\rangle$ we 'chase' I by D into into some $J \in$ SAT(D) and then check if w is in J. A *chase* of I by D is a maximal sequence of distinct relations $I_0, I_1, \ldots$ such that $I = I_0$ and I_{j+1} is obtained from I_j by an application of a *chase rule*. To each tgd in D there corresponds a TT-rule.

TT-rule (for a tgd $\langle w, J\rangle$ in D). I_{j+1} is $\langle w, J\rangle(I_j)$.

Since all the relations in a chase are distinct, it must be a strictly increasing sequence, and we have the following lemma.

Lemma 2.6 ([9]). *All chases of* I *by* D *are finite and have the same final relation, which is in* SAT(D).

[1] Join dependencies are called in [8] *total* join dependencies, and in [17] *full* join dependencies.

[2] Tuple generating dependencies are called *total* tuple generating dependencies in [9, 10, 11].

[3] We use $\subseteq$ to denote set containment and $\subset$ to denote proper containment.

This unique final relation is denoted chase $_D(I)$. It can be used to test implication.

Theorem 2.7 ([9]). *Let D be a set of tgd's, and let $\langle w, I\rangle$ be a tgd. Then $D \models \langle w, I\rangle$ if and only if $w \in \text{chase}_D(I)$.*

Example 1 (*continued*). We show here that $T' \models T$ and $T \models T'$.

To see that $T' \models T$, consider a chase of I by T'. I_0 is I. I_1 is $T'(I)$:

A	B
$a0$	$b1$
$a1$	$b1$
$a1$	$b0$
$a0$	$b0$

The reader can verify that $T'(T'(I)) = T'(I)$, so $\text{chase}_{T'}(I) = I_1$. Since $w \in I_1$, we have $T' \models T$.

To see that T implies T', consider a chase of J by T. J_0 is J.

J_1 is $T(J)$:

A	B
$a0$	$b3$
$a3$	$b3$
$a3$	$b1$
$a1$	$b5$
$a5$	$b5$
$a5$	$b1$
$a1$	$b7$
$a7$	$b7$
$a7$	$b0$
$a0$	$b1$
$a1$	$b1$
$a1$	$b0$

J_2 is $T(J_1) = T(T(J))$:

A	B
$a0$	$b3$
$a3$	$b3$
$a3$	$b1$
$a1$	$b5$
$a5$	$b5$
$a5$	$b1$
$a1$	$b7$
$a7$	$b7$
$a7$	$b0$
$a0$	$b1$
$a1$	$b1$
$a1$	$b0$
$a0$	$b0$

The reader can verify that $T(T(T(J))) = T(T(J))$, so $\text{chase}_T(J) = J_2$. Since $w \in J_2$, we have $T \models T'$.

3. Hilbert-style formal systems

A *Hilbert-style formal system* for a family of dependencies consists of axioms and inference rules. The axioms are schemas of trivial dependencies, e.g., the reflexivity axiom for fd's [2] and mvd's [6]. The inference rules specify whether a dependency is inferrable from some premises, e.g., the transitivity rule for fd's [2] and mvd's

[6]. A *bounded* rule is a rule with a bounded number of premises. A bounded system is a system where all rules are bounded. Let C be a class of dependencies, and let F be a formal system. A *derivation* in C of a dependency $d \in C$ from a set of dependencies $D \subseteq C$ by F is a sequence of dependencies from C: $d_0, d_1, \ldots, d_n$, with $d_n = d$, each of which is either an instance of an axiom of F, a member of D, or is inferrable from earlier d's by one of the inference rules of F. We say that d is *derivable* from D by F in C, denoted $D \vdash_{F,C} d$, if there is a derivation of d from D by F in C. If F and C are understood from context, then we simply write $D \vdash d$. F is *sound* for C if for every $D \subseteq C$ and $d \in C$ we have that $D \vdash_{F,C} d$ entails that $D \models d$; F is *complete* for C if for every $D \subseteq C$ and $d \in C$ we have that $D \models d$ entails that $D \vdash_{F,C} d$. To show that F is sound it suffices to show that, for every d_i in a derivation of d from D in F, $D \models d_i$. That is, if d_i is an instance of an axiom, then it is trivial (proving that the axioms are sound), and if d_i is inferrable from $d_{j_1}, \ldots, d_{j_m}$, then $\{d_{j_1}, \ldots, d_{j_m}\} \models d_i$ (proving that the inference rules are sound).

3.1. Generalized join dependencies

In [11] we presented three systems, called TT_1, TT_2, and TT_3 for tgd's. Essentially, what these system do is simulate the chase. Thus, given a chase $I_0, I_1, \ldots, I_n$ of I by D such that $w \in I_n$, we can construct a derivation by TT_1, TT_2, or TT_3 of $\langle w, I \rangle$ from D. These systems, however, do not specialize to jd's. That is, even when D is a set of jd's and $\langle w, I \rangle$ is a jd, the derivations constructed from the chase may have tgd's that are not jd's. Furthermore, it does not seem possible to simulate the chase by derivations that consists of jd's. We refer the reader to [17] for a discussion of this point.

One may think that the above formal systems for tgd's can solve our motivating problem, that of enumerating all jd's that are implied by a given set of jd's, by generating all tgd's that are implied by the given set of jd's. The difficulty is that a finite set of jd's can imply infinitely many tgd's.

In view of this difficulty, Sciore [17] introduced the class of *generalized join dependencies*. A tgd $\langle w, I \rangle$ is called a generalized join dependency (gjd) if for all $A \in U$ there are at most two repeated A-values in I, and if there are two repeated A-values, then $w[A]$ is one of them.[4] The class of gjd's is denoted by GJD. Clearly, every jd is a gjd, i.e., $JD \subset GJD \subset TTGD$. Note that, for a given universe U, the set GJD is finite. (If $\langle u, I \rangle$ and $\langle v, J \rangle$ are tgd's and h is one-to-one valuation such that $h(u) = v$ and $h(I) = J$, then we say that $\langle u, I \rangle$ and $\langle v, J \rangle$ are *isomorphic*. GJD is finite up to isomorphism of tgd's.)

Sciore then presented a formal system for gjd's that consists of six rules B0–B6. His system is sound. Moreover, he proved that when D is a set of jd's and $\langle w, I \rangle$ is a jd, one can construct a derivation by his system of $\langle w, I \rangle$ from D that consists of gjd's. In this section we improve Sciore's results by showing that a variant of TT_2, which is sound and complete for tgd's, has the same property as Sciore's system.

[4] Generalized join dependencies are called in [17] *full* generalized join dependencies.

Our treatment is not only significantly simpler, but also fits into the larger framework of formal systems for tgd's.

The system TT_2 consists of one axiom and one inference rule.

TTD0' (triviality). $\vdash\langle w, \{w\} \cup I\rangle$.

TTD3 (simplification). $\langle w, I \cup J \cup \{u\}\rangle, \langle u, J\rangle \vdash \langle w, I \cup J\rangle$.

Theorem 3.1 ([11]). *The system* TT_2 *is sound and complete for tgd's.*

The system TT_2' is a variant of TT_2.

TTD0 (triviality). $\vdash\langle w, \{w\}\rangle$.

TTD1 (covering). $\langle u, I\rangle \vdash \langle v, J\rangle$ if $\langle v, J\rangle \leqslant \langle u, I\rangle$.

TTD3' (simplification). $\langle w, I \cup J \cup \{u\}\rangle, \langle v, J\rangle \vdash \langle w, I \cup J\rangle$, if, for some $X \subseteq U$, $u[\bar{X}]$ is nonrepeated in $I \cup J \cup \{u, w\}$ and $u[X] = v[X]$.

Rules TTD0' and TTD1 generalize the triviality axiom and the covering rule for jd's in [8, 18] and also generalize rules B0, B1, B2, and B5 for gjd's in [17]. Rules TTD3 and TTD3', however, have no analogue in [8, 17, 18].

Example 3. Let $U = ABCD$. Let $\langle v, J\rangle$ be

	A	B	C	D
v:	$a0$	$b0$	$c0$	$d0$
	$a0$	$b0$	$c1$	$d0$
	$a0$	$b1$	$c0$	$d1$

Let $\langle w, I \cup J \cup \{u\}\rangle$ be

	A	B	C	D
w:	$a2$	$b0$	$c0$	$d0$
	$a2$	$b0$	$c2$	$d0$
	$a0$	$b0$	$c1$	$d0$
	$a0$	$b1$	$c0$	$d1$
u:	$a1$	$b0$	$c0$	$d2$

Let $X = BC$. We have that $u[AD]$ is nonrepeated in $I \cup J \cup \{u, w\}$ and $u[BC] = v[BC]$.

By rule TTD3', $\langle w, I \cup J \cup \{u\}\rangle$, $\langle v, J\rangle \vdash \langle w, I \cup J\rangle$, where $\langle w, I \cup J\rangle$ is

	A	B	C	D
w:	$a2$	$b0$	$c0$	$d0$
	$a2$	$b0$	$c2$	$d0$
	$a0$	$b0$	$c1$	$d0$
	$a0$	$b1$	$c0$	$d1$

Theorem 3.2. *The system* TT_2 *is sound and complete for tgd's.*

Proof. *Soundness*: Rules TTD0, TTD1, and TTD3 are shown to be sound in [11]. We now show that rule TTD3' is also sound. Suppose that $u[\bar{X}]$ is nonrepeated in $I \cup J \cup \{u, w\}$ and $u[X] = v[X]$. Define a valuation h such that $h(u[\bar{X}]) = v[\bar{X}]$ and h is the identity elsewhere. We now have $h(w) = w$ and $h(I \cup J \cup \{u\}) = I \cup J \cup \{v\}$. Thus, by rule TTD1, $\langle w, I \cup J \cup \{u\}\rangle \models \langle w, I \cup J \cup \{v\}\rangle$. Now, by rule TTD3, $\langle w, I \cup J \cup \{v\}\rangle$, $\langle v, J\rangle \models \langle w, I \cup J\rangle$.

Completeness: It suffices to show that the rules of TT_2' imply the rules of TT_2. Since $\langle w, \{w\}\rangle \leqslant \langle w, \{w\} \cup I\rangle$, rules TTD0 and TTD1 together imply rule TTD0'. Rule TTD3 is a special case of rule TTD3' by taking X to be U. □

Theorem 3.3. *Let D be a set of jd's, and let d be a jd. If $D \models d$, then $D \vdash_{\mathrm{TT}_2', \mathrm{GJD}} d$.*

Proof. Let D be a set of jd's, let $\langle w, I\rangle$ be a jd, and suppose that $D \models \langle w, I\rangle$. By Theorem 2.7, it suffices to show that for every $u \in \mathrm{chase}_D(I)$ we have $D \vdash_{\mathrm{GJD}} \langle u, I\rangle$. (Note that $\langle v, I\rangle$ is a gjd for any tuple v such that $\langle v, I\rangle$ is a tgd, because there is at most one repeated A-value in I for all $A \in U$.) Let $I_0, I_1, \ldots, I_n$ be a chase of I by D. We show by induction on j that, for every $u \in I_j$, $D \vdash_{\mathrm{GJD}} \langle u, I\rangle$. I_0 is I, so if $u \in I$, then $\vdash_{\mathrm{GJD}} \langle u, \{u\}\rangle$ by rule TTD0, and $\langle u, \{u\}\rangle \vdash_{\mathrm{GJD}} \langle u, I\rangle$ by rule TTD1.

Suppose now that the assumption holds for I_j and let $u \in I_{j+1}$. That is, there is a tgd $\langle v, J\rangle \in D$ such that $u \in \langle v, J\rangle(I_j)$. Let $\langle v, J\rangle$ represent the jd $*[\boldsymbol{R}]$, $\boldsymbol{R} = \{R_1, \ldots, R_m\}$. Construct a tgd $\langle u, K\rangle$, $K = \{u_1, \ldots, u_m\}$, which also represents $*[\boldsymbol{R}]$. It is easy to define a one-to-one valuation h such that $h(J) = K$ and $h(v) = u$. Thus $\langle v, J\rangle \vdash_{\mathrm{GJD}} \langle u, K\rangle$ by rule TTD1. Also, by rule TTD1, $\langle u, K\rangle \vdash \langle u, K \cup I\rangle$. We claim that $\langle u, K \cup I\rangle$ is a gjd. In proof, note that for all $A \in U$ there is at most one repeated A-value in I and at most one repeated A-value in K, that if $t[A]$ is nonrepeated in K, then $t[A]$ is not in $\pi_A(I)$, and that if $t[A]$ is repeated in K, then $u[A] = t[A]$. Since $\langle u, K\rangle$ represents $*[\boldsymbol{R}]$, we have $u \in \langle u, K\rangle(I_j)$. That is, there is a valuation h on K such that $h(u) = u$ and $h(K) \subseteq I_j$. Let $t_i = h(u_i) \in I_j$, $1 \leqslant i \leqslant m$. By the induction hypothesis, $D \vdash_{\mathrm{GJD}} \langle t_i, I\rangle$, $1 \leqslant i \leqslant m$. Now $u_i[\bar{R}_i]$ is nonrepeated in $K \cup I \cup \{u\}$ and

$$t_i[R_i] = h(u_i[R_i]) = h(u[R_i]) = u[R_i] = u_i[R_i],$$

so by m applications of rule TTD3', $\langle u, K \cup I\rangle$, $\langle t_1, I\rangle, \ldots, \langle t_m, I\rangle \vdash_{\mathrm{GJD}} \langle u, I\rangle$. □

3.2. Extended join dependencies

The results of the previous section can be improved in two directions. First, we can restrict the class of dependencies that have to be considered in order to enumerate jd's. Secondly, we can prove that the formal system used is complete for this class of dependencies, unlike the system TT_2' that is not known to be complete for gjd's. The price to pay for these improvements is having to deal with unbounded inference rules.

We consider here the class of *extended join dependencies*. A tgd $\langle w, I\rangle$ is called an extended join dependency (xjd) if for all $A \in U$ there is at most one repeated A-value in I.[5] The class of xjds is denoted by XJD. Clearly, every jd is an xjd, and every xjd is a gjd, i.e., $\mathrm{JD} \subset \mathrm{XJD} \subset \mathrm{GJD} \subset \mathrm{TTGD}$.

We use the system TT_3 from [11].

TTD0′ (triviality). $\vdash \langle w, \{w\} \cup I\rangle$.

TTD4 (transitivity). $\langle w, I\rangle, \langle u_1, J\rangle, \ldots, \langle u_m, J\rangle \vdash \langle u, J\rangle$ if $u \in \langle w, I\rangle(\{u_1, \ldots, u_m\})$.

Rule TTD4 is unbounded because it may have an unbounded number of premises.

Theorem 3.4 ([11]). *The system* TT_3 *is sound and complete for tgd's.*

Let $\langle u, J\rangle$ be a jd. Then every tgd $\langle v, J\rangle$ is an xjd, because there is at most one repeated A-value in I for all $A \in U$, but not necessarily a jd, because $v[A]$ may not be the repeated A-value for some $A \in U$. Thus, from Theorem 2.7, there is a one-to-one correspondence between the tuples of $\mathrm{chase}_D(J)$ and the xjd's $\langle v, J\rangle$ implied by D. Using this characterization, we can show that the system TT_3 is complete for xjd's.

Theorem 3.5. *The system* TT_2 *is complete for xjd's.*

Proof. Let D be a set of xjd's, let $\langle v, J\rangle$ be an xjd, and suppose that $D \models \langle v, J\rangle$. By Theorem 2.7 it suffices to show that, for every $u \in \mathrm{chase}_D(J)$, $D \vdash_{\mathrm{XJD}} \langle u, J\rangle$. (Note that $\langle u, I\rangle$ is an xjd.) Let $J_0, \ldots, J_n$ be a chase of J by D. We show by induction on i that, for every $u \in J_i$, $D \vdash_{\mathrm{XJD}} \langle u, J\rangle$. J_0 is J so if $u \in J$, then $D \vdash_{\mathrm{XJD}} \langle u, J\rangle$ by rule TTD0′. Suppose now that the assumption holds for $J_i = \{u_1, \ldots, u_m\}$. Let $u \in J_{i+1}$. That is, there is an xjd $\langle w, I\rangle \in D$ such that $u \in \langle w, I\rangle(\{u_1, \ldots, u_m\})$. By the induction hypothesis, $D \vdash_{\mathrm{XJD}} \langle u_k, J\rangle$, for $1 \leq k \leq m$, so $D \vdash_{\mathrm{XJD}} \langle u, J\rangle$ by rule TTD4. □

[5] That is, if $\langle w, I\rangle$ is what Aho et al. [3] call a *simple tableau.*

4. A Gentzen-style formal system for jd's

When $\langle u, J\rangle$ is a jd, every tgd $\langle v, J\rangle$ is an xjd. Using the correspondence between the tuples of $\text{chase}_D(I)$ and the xjd's $\langle v, J\rangle$ implied by D, we showed that the system TT_3 is complete for xjd's. The difficulty in obtaining a formal systems for jd's stems from the fact that not all tuples in $\text{chase}_D(I)$ correspond to jd's.

To study the situation in detail, we need some more machinery. From now on we treat asc's as sequences of attribute sets rather than unordered collections. Thus an asc is a sequence $(R_1, \ldots, R_m)$ of attribute sets such that $\bigcup_{i=1}^{m} R_i = U$. For an asc $\boldsymbol{R} = (R_1, \ldots, R_m)$, we partition the attributes of U into two sets:

$$\text{MANY}(\boldsymbol{R}) = \{A: \text{for some } 1 \leq i, j \leq m,\ i \neq j,\ A \in R_i \cap R_j\},$$

$$\text{ONCE}(\boldsymbol{R}) = \{A: \text{for all } 1 \leq i, j \leq m, \text{ if } i \neq j, \text{ then } A \notin R_i \cap R_j\}.$$

That is, $\text{MANY}(\boldsymbol{R})$ is the set of attributes that belong to at least two elements of $\boldsymbol{R}$ and $\text{ONCE}(\boldsymbol{R})$ is the set of attributes that belong to exactly one element of $\boldsymbol{R}$. For every $R \in \boldsymbol{R}$, define the *stem* of R: $\text{ST}(R) = R \cap \text{MANY}(\boldsymbol{R})$. I satisfies $*[\boldsymbol{R}]$ if whenever there are tuples $w_1, \ldots, w_m$ in I such that $w_i[R_i \cap R_j] = w_j[R_i \cap R_j]$, then there is in I a tuple w such that $w[R_i] = w_i[R_i]$. Thus attributes in $\text{MANY}(\boldsymbol{R})$ and $\text{ONCE}(\boldsymbol{R})$ play different roles in the 'meaning' of $*[\boldsymbol{R}]$.

Let $T_{\boldsymbol{R}}$ be $\langle u, J\rangle$, $J = \{u_1, \ldots, u_m\}$. Suppose $\langle v, J\rangle$ is a tgd such that $v[\text{MANY}(\boldsymbol{R})] = u[\text{MANY}(\boldsymbol{R})]$. Then $\langle v, J\rangle$ represents the jd $*[S_1, \ldots, S_m]$, where $S_i = \{A: v[A] = u_i[A]\}$. (Note that $\text{ST}(S_i) = \text{ST}(R_i)$.) Conversely, if $*[S_1, \ldots, S_m]$ is a jd implied by D such that $\text{ST}(S_i) = \text{ST}(R_i)$, $1 \leq i \leq m$, then there is a tuple v in $\text{chase}_D(I)$ such that $\langle v, J\rangle$ represents $*[S_1, \ldots, S_m]$ and $v[\text{MANY}(\boldsymbol{R}) = u[\text{MANY}(\boldsymbol{R})]$. We say that $*[S_1, \ldots, S_m]$ has the same *stem sequence* as $*[\boldsymbol{R}]$. Thus there is a one-to-one correspondence between the tuples of $\text{chase}_D(J)$ with the same $\text{MANY}(\boldsymbol{R})$-value as u and the jd's with the same stem sequence as $*[\boldsymbol{R}]$ that are implied by D.

In order to simulate the chase, we have to associate a jd $*[\boldsymbol{S}^v]$ with each tuple $v \in \text{chase}_D(I)$. $*[\boldsymbol{S}^v]$, however, does not carry the same information as v. In order to keep the same information in the derivation, we also have to carry with us the *stem basis*, which is a generalization of the stem sequence. A stem basis $\boldsymbol{X}$ is a sequence of attributes sets $(X_1, \ldots, X_m)$ such that if $A \in X_i$, then $A \in X_j$, for some $j \neq i$. Note that if $m = 1$, then $\boldsymbol{X} = (\emptyset)$. A jd $*[\boldsymbol{R}] = *[R_1, \ldots, R_m]$ is $\boldsymbol{X}$-based if $X_i \subseteq R_i$, and $(R_i - X_i) \cap (R_j - X_j) = \emptyset$ for $i \neq j$, $1 \leq i, j \leq m$.

If we try to specialize rule TTD4 to jd's, we realize that the rule is not sound unless all the premises of the rule are jd's with the same stem basis. That means that the concatenation of two sound derivations is not necessarily a sound derivation, because the jd's in the two derivations may have different stem bases. The ability to concatenate derivations is, however, a basic feature of Hilbert-style systems. The solution is to revert to Gentzen-style formal systems, which deals with *sequents* instead of dependencies. (See [14] for a description of a Gentzen-style formal system for first-order logic.) A *sequent* is an expression $\boldsymbol{X}: D \rightarrow d$, where $\boldsymbol{X}$ is a stem basis,

D is a finite set of jd's and d is an $\boldsymbol{X}$-based jd. $\boldsymbol{X}$ is the *label* of the sequent, D is the *antecedent*, and d is the *succedent*.

The interpretation of $\rightarrow$ is that of implication. Namely, $\boldsymbol{X}: D \rightarrow d$ is true if $D \models d$. The label is needed to guide the derivations. The inference rules are such that sequents with different labels can not interact. A Gentzen-style formal system F has axioms and inference rules for sequents rather than dependencies. F is sound if whenever $\boldsymbol{X}: D \rightarrow *[\boldsymbol{R}]$ is derivable by F for some stem basis $\boldsymbol{X}$, we have $D \models *[\boldsymbol{R}]$. F is complete if whenever $D \models *[\boldsymbol{R}]$, the sequent $\boldsymbol{X}: D \rightarrow d$ is derivable by F for some stem basis $\boldsymbol{X}$.

We now present the Gentzen-style system J.

ZJD0. $\vdash \boldsymbol{X}: D \rightarrow *[X_1, \ldots, X_{i-1}, U, X_{i+1}, \ldots, X_m]$ for a stem basis $\boldsymbol{X} = (X_1, \ldots, X_m)$, $1 \leq i \leq m$.

ZJD1. Let $*[R_1, \ldots, R_k] \in D$, let $\boldsymbol{X} = (X_1, \ldots, X_m)$ be a stem basis, and let $*[\boldsymbol{S}^i] = *[S_1^i, \ldots, S_m^i]$ be $\boldsymbol{X}$-based jd's such that $R_i \cap R_j \cap S_p^i \subseteq S_p^j$ for all $1 \leq i, j \leq k$, $1 \leq p \leq m$. Then

$$\boldsymbol{X}: D \rightarrow *[\boldsymbol{S}^1], \ldots, \boldsymbol{X}: D \rightarrow *[\boldsymbol{S}^k] \vdash \boldsymbol{X}: D \rightarrow *[Q_1, \ldots, Q_m],$$

where $Q_i = \bigcup_{j=1}^{k} (R_j \cap S_i^j)$.

Example 4. Let $U = ABCD$, $D = \{*[\boldsymbol{R}]\}$, $\boldsymbol{R} = \{ABC, AD\}$, $\boldsymbol{X} = \{AB, AC, BC\}$, $\boldsymbol{S}^1 = \{ABCD, AC, BC\}$, and $\boldsymbol{S}^2 = \{AB, ABCD, BC\}$. The reader can verify that $\boldsymbol{X}$ is a stem basis, and that $*[\boldsymbol{S}^1]$ and $*[\boldsymbol{S}^2]$ are $\boldsymbol{X}$-based. Furthermore, the conditions of rule ZJD1 are satisfied. Consequently,

$$\boldsymbol{X}: D \rightarrow *[\boldsymbol{S}_1], \boldsymbol{X}: D \rightarrow *[\boldsymbol{S}_2] \vdash \boldsymbol{X}: D \rightarrow *[\boldsymbol{Q}],$$

where $\boldsymbol{Q} = \{ABC, ACD, BC\}$.

Theorem 4.1. *The system J is sound and complete for jd's.*

Proof. *Soundness*: We first show that the succedent is always $\boldsymbol{X}$-based.

ZJD0: $*[X_1, \ldots, X_{i-1}, U, X_{i+1}, \ldots, X_m]$ is clearly $\boldsymbol{X}$-based.

ZJD1: We show that $*[\boldsymbol{Q}] = *[Q_1, \ldots, Q_m]$ is $\boldsymbol{X}$-based. Let $A \in X_p$. Then $A \in S_p^i$ for all $1 \leq i \leq k$, so $A \in Q_p$. Assume now that $A \in (Q_p - X_p) \cap (Q_q - X_q)$ for $q \neq p$. That is, for some i, j we have

$$A \in R_i \cap S_p^i \cap R_j \cap S_q^j \cap \bar{X}_p \cap \bar{X}_q.$$

But

$$R_i \cap R_j \cap S_p^i \cap \bar{X}_p \subseteq S_p^j \cap \bar{X}_p,$$

so

$$A \in (S_p^j - X_p) \cap (S_q^j - X_q);$$

a contradiction. It follows that $*[\boldsymbol{Q}]$ is $\boldsymbol{X}$-based.

For a stem basis $\boldsymbol{X} = (X_1, \ldots, X_m)$, construct a relation $I_{\boldsymbol{X}} = \{w_1, \ldots, w_m\}$ such that $w_i[A] = w_j[A]$ iff $A \in X_i \cap X_j$. Let $*[\boldsymbol{Q}]$ be an $\boldsymbol{X}$-based jd. We define a tuple $w_{\boldsymbol{Q}}$ as follows. If $A \in Q_i - X_i$ for some i, then $w_{\boldsymbol{Q}}[A] = w_i[A]$. else $w_{\boldsymbol{Q}}[A] = w_i[A]$ for some i such that $A \in Q_i$. $w_{\boldsymbol{Q}}$ is well defined because if $A \in Q_i - X_i$, then for no $j \neq i$ is $A \in Q_j - X_j$. Otherwise, whenever $A \in Q_i$ also $A \in X_i$, so if $A \in Q_i \cap Q_j$, then $A \in X_i \cap X_j$ and $w_i[A] = w_j[A]$.

Example 4 (*continued*). Let $I_{\boldsymbol{X}} = \{w_1, w_2, w_3\}$ be

	A	B	D	D
w_1:	$a0$	$b0$	$c1$	$d1$
w_2:	$a0$	$b1$	$c0$	$d2$
w_3:	$a1$	$b0$	$c0$	$d3$

Now $w_{\boldsymbol{Q}}$ is the tuple

A	B	D	D
$a0$	$b0$	$c0$	$d2$

Proof of Theorem 4.1 (*continued*). We show by induction on the length of the derivation that if $\vdash_J \boldsymbol{X}: D \to *[\boldsymbol{Q}]$, then $w_{\boldsymbol{Q}} \in \text{chase}_D(I_{\boldsymbol{X}})$.

ZJD0: $*[\boldsymbol{Q}]$ is $*[X_1, \ldots, X_{i-1}, U, X_{i+1}, \ldots, X_m]$. Here $w_{\boldsymbol{Q}} = w_i \in I_{\boldsymbol{X}}$.

ZJD1: $*[\boldsymbol{Q}]$ is $*[Q_1, \ldots, Q_m]$, $Q_i = \bigcup_{j=1}^{k} (R_j \cap S_i^j)$. Let t_i denote $w_{\boldsymbol{S}^i}$. By the induction hypothesis, $t_i \in \text{chase}_D(I_{\boldsymbol{X}})$, $1 \leq i \leq k$. Let t be the tuple defined by $t[R_i] = t_i[R_i]$, $1 \leq i \leq k$. We have to show that t is well defined, that is, $t_i[A] = t_j[A]$ if $A \in R_i \cap R_j$. $t_i[A] = w_p[A]$ for some p such that $A \in S_p^i$, and $t_j[A] = w_q[A]$ for some q such that $A \in S_q^j$. If $w_p[A] \neq w_q[A]$, then $A \in S_p^i - X_p$ or $A \in S_q^j - X_q$. Assume without loss of generality that $A \in S_p^i - X_p$. But

$$R_i \cap R_j \cap (S_p^i - X_p) \subseteq S_p^j - X_p,$$

so $A \in S_p^j - X_p$ and $t_j[A] = w_p[A]$—a contradiction. It follows that t is well defined.

Since $\text{chase}_D(I_{\boldsymbol{X}})$ is in $\text{SAT}(D)$, we have that

$$m_{\boldsymbol{R}}(\text{chase}_D(I_{\boldsymbol{X}})) = \text{chase}_D(I_{\boldsymbol{X}}),$$

so $t \in \text{chase}_D(I_{\boldsymbol{X}})$. It remains to show that t is exactly $w_{\boldsymbol{Q}}$.

Suppose first that $A \in Q_p - X_p$. Then, for some i, $A \in R_i \cap S_p^i \cap \bar{X}_p$. It follows that $t[A] = t_i[A] = w_p[A]$. Suppose now whenever $A \in Q_p$ also $A \in X_p$, and that $A \in Q_q$ for some q. Then $A \in R_i \cap S_q^i$ for some i, and $t[A] = t_i[A]$. If $t_i[A] \neq w_q[A]$, then, for some p, $A \in S_p^i - X_p$, so $A \in Q_p - X_p$—a contradiction.

We have shown that if $\vdash_J \boldsymbol{X}: D \to *[\boldsymbol{Q}]$, then $w_{\boldsymbol{Q}} \in \text{chase}_D(I_{\boldsymbol{X}})$, so, by Theorem 2.7, $D \models \langle w_{\boldsymbol{Q}}, I_{\boldsymbol{X}} \rangle$. To complete the soundness proof, we have to show that $\langle w_{\boldsymbol{Q}}, I_{\boldsymbol{X}} \rangle \leq T_{\boldsymbol{Q}}$. Let $T_{\boldsymbol{Q}}$ be $\langle v, J \rangle$, where $J = \{v_1, \ldots, v_m\}$. Define a valuation h such that $h(w_i) = v_i$ for $1 \leq i \leq m$. h is well defined, because if $w_i[A] = w_j[A]$, then $A \in X_i \cap X_j$, so $A \in Q_i \cap Q_j$ and $v_i[A] = v_j[A]$. If $w_{\boldsymbol{Q}}[A] = w_p[A]$, then $A \in Q_p$ and $v[A] = v_p[A]$. Therefore, $h(w_{\boldsymbol{Q}}[A]) = v[A]$. It follows that $h(w_{\boldsymbol{Q}}) = v$.

Completeness: Suppose that $D \models *[\boldsymbol{Q}]$, $\boldsymbol{Q} = \{Q_1, \ldots, Q_m\}$, $T_{\boldsymbol{Q}} = \langle w, I \rangle$, $I = \{w_1, \ldots, w_m\}$. Define the stem basis $\boldsymbol{X} = (\text{ST}(Q_1), \ldots, \text{ST}(Q_m))$. It is easy to see that we can take $I_{\boldsymbol{X}}$ to be I. By Theorem 2.7, $w \in \text{chase}_D(I)$. With each tuple u in $\text{chase}_D(I)$ we associate a jd $*[\boldsymbol{S}^u] = *[S_1^u, \ldots, S_m^u]$, where $S_i^u = X_i \cup \{A: u[A] = w_i[A]\}$. We claim that $*[\boldsymbol{S}^u]$ is $\boldsymbol{X}$-based. Clearly, $X_i \subseteq S_i^u$, and if

$$A \in (S_i^u - X_i) \cap (S_j^u - X_j),$$

then $w_i[A] = w_j[A]$—a contradiction, because $w_i[A] = w_j[A]$ iff $A \in X_i \cap X_j$. Observe that if $u[\text{MANY}(\boldsymbol{Q})] = w[\text{MANY}(\boldsymbol{Q})]$, then $*[\boldsymbol{S}^u]$ is equivalent to $\langle u, I \rangle$.

Let the chase of I by D be $I_0, \ldots, I_n$. We show by induction on j that for every $u \in I_j$, we have $\vdash_J \boldsymbol{X}: D \to *[\boldsymbol{S}^u]$.

Basis $(j = 0)$: I_0 is I, so u is w_i for some i and $*[\boldsymbol{S}^u]$ is $*[X_1, \ldots, X_{i-1}, U, X_{i+1}, \ldots, X_m]$. By ZJD0, $\vdash_J \boldsymbol{X}: D \to *[\boldsymbol{S}^u]$.

Induction: Let $u \in I_{j+1}$. There are a jd $*[R_1, \ldots, R_k] \in D$ and tuples $u_1, \ldots, u_k \in I_j$ such that $u[R_i] = u_i[R_i]$, $1 \leq i \leq k$. Let $*[\boldsymbol{S}^i]$ denote $*[\boldsymbol{S}^{u_i}]$. That is,

$$S_p^i = X_p \cup \{A: u_i[A] = w_p[A]\}.$$

By the induction hypothesis, $\vdash \boldsymbol{X}: D \to *[\boldsymbol{S}^i]$. With u we associate $*[\boldsymbol{S}^u] = *[S_1^u, \ldots, S_m^u]$, where

$$S_p^u = X_p \cup \{A: u[A] = w_p[A]\} = X_p \cup \bigcup_{h=1}^{k} \{A \in R_h: u[A] = w_p[A]\}$$

$$= X_p \cup \bigcup_{h=1}^{k} \{A \in R_h: u_h[A] = w_p[A]\} = X_p \cup \bigcup_{h=1}^{k} (R_h \cap S_p^h)$$

$$= \bigcup_{h=1}^{k} (R_h \cap S_p^h).$$

To prove that $\vdash_J \boldsymbol{X}: D \to *[S^u]$ by rule ZJD1, we have to show that

$$R_f \cap R_g \cap S_p^f \subseteq S_p^g.$$

We have $X_p \subseteq S_p^f$ and $X_p \subseteq S_p^g$. Let

$$A \in R_f \cap R_g \cap (S_p^f - X_p).$$

Then $u[A] = u_f[A] = w_p[A]$ and $u[A] = u_g[A]$, so $u_g[A] = w_p[A]$, and $A \in S_p^g - X_p$.

In particular, $\vdash_J \boldsymbol{X}: D \to *[S^w]$. But $*[S^w]$ is just $*[\boldsymbol{Q}]$, which completes the proof. □

5. Concluding remarks

In this paper we have investigated whether a sound and complete formal system for join dependencies can be found. We have shown a bounded formal system that is strong enough to derive join dependencies using only generalized join dependencies in the derivation and an unbounded formal system that is complete for extended join dependencies. Both systems are also sound and complete for tuple generating dependencies. We have also constructed a sound and complete unbounded Gentzen-style system for join dependencies.

Several problems remain open:

(1) Is the system TT_2' complete for any subclass of TGD that contains JD?

(2) Is there a sound and complete bounded formal system for extended join dependencies?

(3) Is there a sound and complete bounded Gentzen-style system for join dependencies?

Finally, we would like to comment about the usefulness of the system *J*. As was observed in Section 1, the formal system consisting of the rule

$$d_1, \ldots, d_k \vdash d \quad \text{if} \{d_1, \ldots, d_k\} \models d$$

is sound and complete for every class of dependencies for which the implication problem is solvable. The interest in 'elegant' systems is twofold. First, such a system can often lead to the construction of efficient algorithms for testing implication, as the formal system for functional dependencies of [2] leads to the efficient algorithm of [4], and the formal system for multivalued dependencies of [6] leads to the efficient algorithm of [5]. Furthermore, such systems offer more insight into the properties of the class of dependencies under study and facilitate the use of dependencies in the design of the database schema. In our view, the system *J* is too complex to offer the second advantage. Nevertheless, the system offer a syntactic description of the chase, and this description may make it possible to construct a subexponential algorithm for testing implication of join dependencies.

Acknowledgment

We are grateful to Ed Sciore for his extremely helpful comments on earlier drafts of this paper.

References

[1] A.V. Aho, C. Beeri and J.D. Ullman, The theory of joins in relational data-bases, *ACM Trans. Database Systems* **4** (1979) 297-314.

[2] W.W. Armstrong, Dependency structure of database relationships, *Proc. IFIP 74* (North-Holland, Amsterdam, 1974) 580-583.

[3] A.V. Aho, Y. Sagiv and J.D. Ullman, Equivalence among relational expressions, *SIAM J. Comput.* **8** (1979) 218-246.
[4] C. Beeri and P.A. Bernstein, Computational problems related to the design of normal form relational schemas, *ACM Trans. Database Systems* **4** (1979) 30-59.
[5] C. Beeri, On the membership problem for multivalued dependencies, *ACM Trans. Database Systems* **5** (1980) 241-259.
[6] C. Beeri, R. Fagin and J.H. Howard, A complete axiomatization for functional and multivalued dependencies in database relations, *Proc. ACM Conf. on Management of Data* (1977) 47-61.
[7] C. Beeri, A.O. Mendelzon, Y. Sagiv and J.D. Ullman, Equivalence of relational database schemes, *SIAM J. Comput.* **10** (1981) 647-656.
[8] C. Beeri and M.Y. Vardi, On the properties of join dependencies, in: H. Gallaire, J. Minker and J.M. Nicolas, eds., *Advances in Database Theory* (Plenum, New York, 1981) 25-72.
[9] C. Beeri and M.Y. Vardi, A proof procedure for data dependencies, *J. ACM* **31** (4) (1984) 718-741.
[10] C. Beeri and M.Y. Vardi, On the complexity of testing implications of data dependencies, Rept., Dept. of Comput. Sci., The Hebrew Univ. of Jerusalem, 1980.
[11] C. Beeri and M.Y. Vardi, Formal systems for tuple and equality generating dependencies, *SIAM J. Comput.* **13** (1) (1984) 76-98.
[12] E.F. Codd, Relational completeness of database sublanguages, in: R. Rustin, ed., *Data Base Systems* (Prentice-Hall, Englewood Cliffs, NJ, 1972) 65-98.
[13] E.F. Codd, Further normalization of the database relational model, in: R. Rustin, ed., *Data Base Systems* (Prentice-Hall, Englewood Cliffs, NJ, 1972) 33-64.
[14] S.C. Kleene, *Mathematical Logic* (Wiley, New York, 1967).
[15] D. Maier, A.O. Mendelzon and Y. Sagiv, Testing implications of data dependencies, *ACM Trans. Database Systems* **2** (1977) 201-222.
[16] J. Rissanen, Theory of relations for databases—a tutorial survey, *Proc. 7th Symp. on Mathematical Foundations of Computer Science* (1978) 537-551.
[17] E. Sciore, A complete axiomatization of full join dependencies, *J. ACM* **29** (2) (1982) 373-393.
[18] M.Y. Vardi, Axiomatization of functional and join dependencies in the relational model, M.Sc. Thesis, The Weizmann Institute of Science, 1980.